FONT DESIGNING

ICS535–Computer Graphics Course

Final Term Project Report

Ahmed Abdul Mujeeb Qhusro

220102

&

Mirza Humayun Baig

220318

Submitted to

Dr. M. Sarfraz

on

June 3, 2003

Department of Information and Computer Science
College of Computer Science and Engineering
King Fahd University of Petroleum and Minerals
Dhahran - Kingdom of Saudi Arabia
2003

Abstract

Two fundamental techniques of Font representations are Bitmap and Outline. The criteria of font designing using approximating curves are to reduce the memory requirements which are high using bitmap files. There are many other drawbacks of representing images using bitmap files, which we discuss in the report later. Our aim here is to extract the digital image and convert it to outline image for open curves, but here we have drawn the curve using some curve drawing technique and experimented instead of scanning and taking the digitized image. We extracted the boundary points of the curve, and then we determined the corner points using Corner Detection algorithm discussed in section 2. Using Least Square method, we fitted the parametric curve to each segment. If the fitted curve did not match we determined the significant points using Break point concept discussed in the report till we get the required curve fit.

1.Introduction

There are two fundamental techniques for storing fonts in computer is Bitmap and Outline. Number of pixels determines the number of shades of gray or color that can be represented. For example one pixel per bit image has only two possible values of bit i.e. bit is either on (1) or off(0)the bitmaps for font cache are usually created by scanning in enlarged pictures of characters in various sizes. Generating a character requires to copy the pixel values from font cache into the frame buffer and then on to the screen at the desired position.

Fonts can be represented by outline using Spines. Spines are piecewise polynomial parametric curves, generated by varying a parameter over a specified range. Piecewise polynomial functions provides smooth curves. These Spines are used to store the outline of characters. Although each definition may take more space than its representation in a font cache but multiple sizes may be derived from a single stored representation by suitable scaling. Different types of faces can also be derived. Another advantage of storing characters in a device independent form is that the outlines may be arbitrarily translated, rotated, scaled and clipped.

Drawbacks of bitmap representation of fonts

1. It requires a distinct font cache for each combination of font, size and face for each different resolution of display or output device

2. Bitmap representation of font behaves unpredictably when subject to affine transformations, there by giving unacceptable shape and resolution
3. Bitmap fonts always appear the same even if the resolution of the device is enhanced.

2.Corner Point Detection and Curve Fitting

The fundamental objective behind our system is to obtain the parametric representation of character that matches with the digitized image. Digitized image of a character can be obtained directly from some electronic device or by scanning an image. The contour of the digitized image is extracted by using some boundary detection algorithm. The algorithm returns the number of pieces for each piece number of boundary points. The points obtained by extracting the contour of digitized image may have noise(jagged Edges).Filtering is done to eliminate the jaggies.We have taken open curve shown in figure 1 and applied the approximating method.

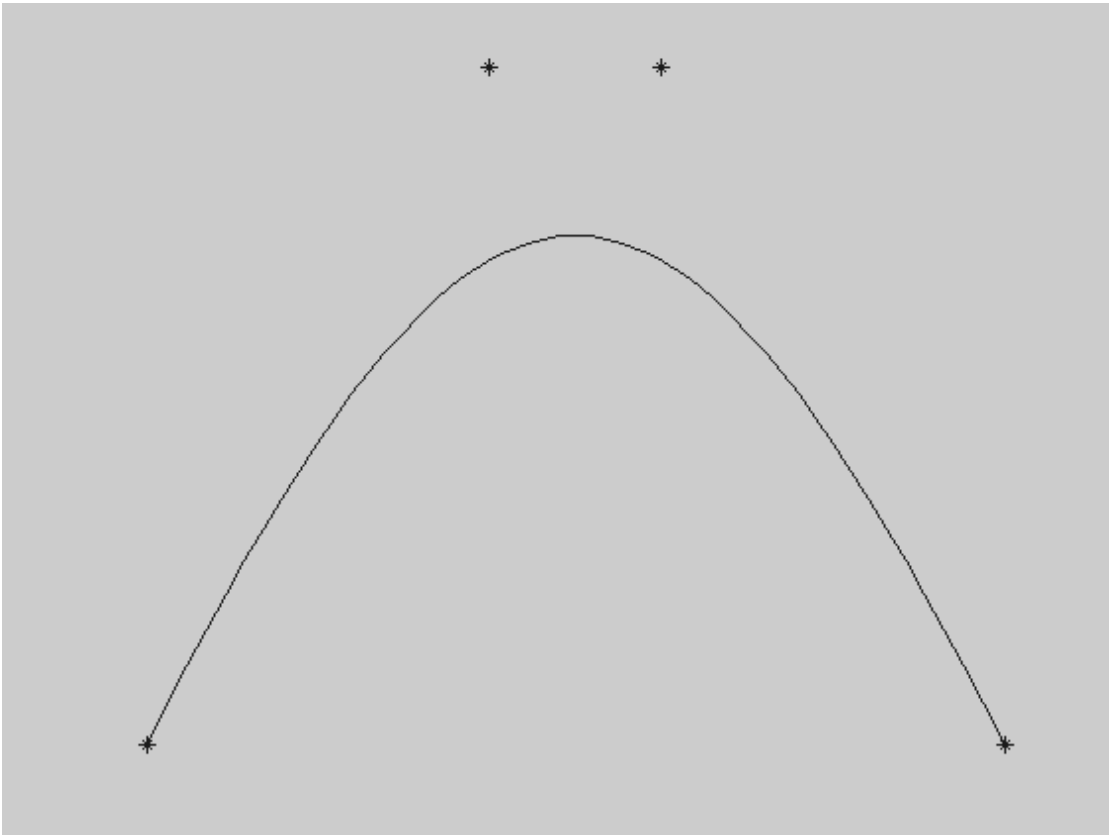


Figure 1: Bezier Curve

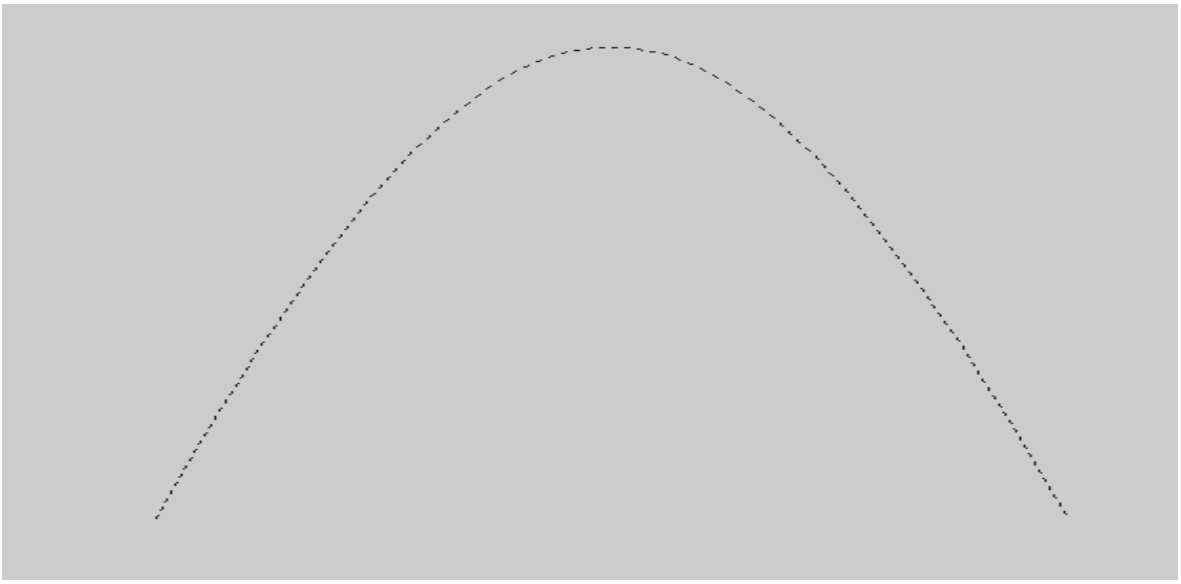


Figure 2: Digitized Curve

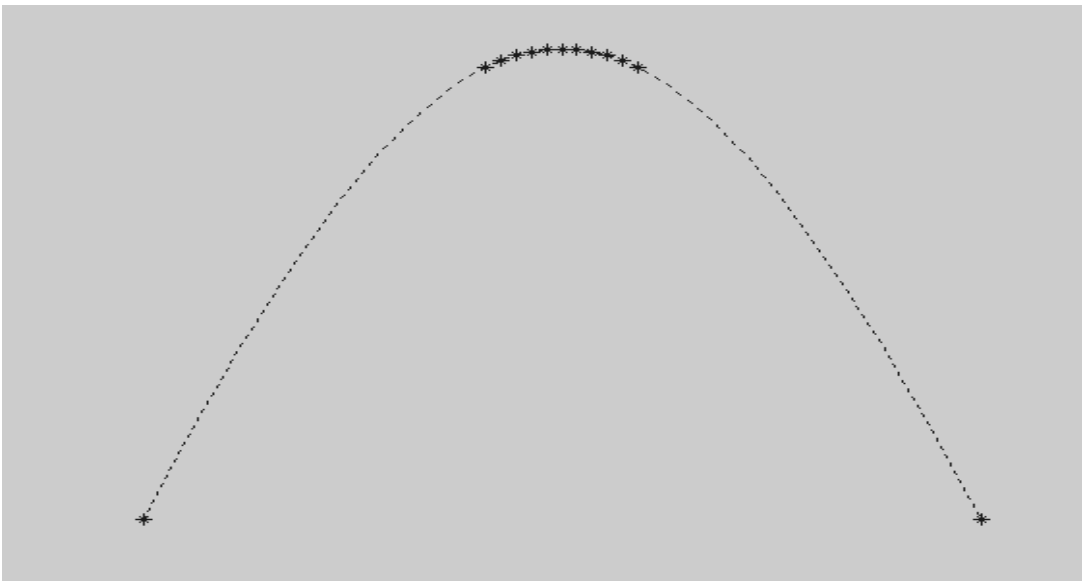
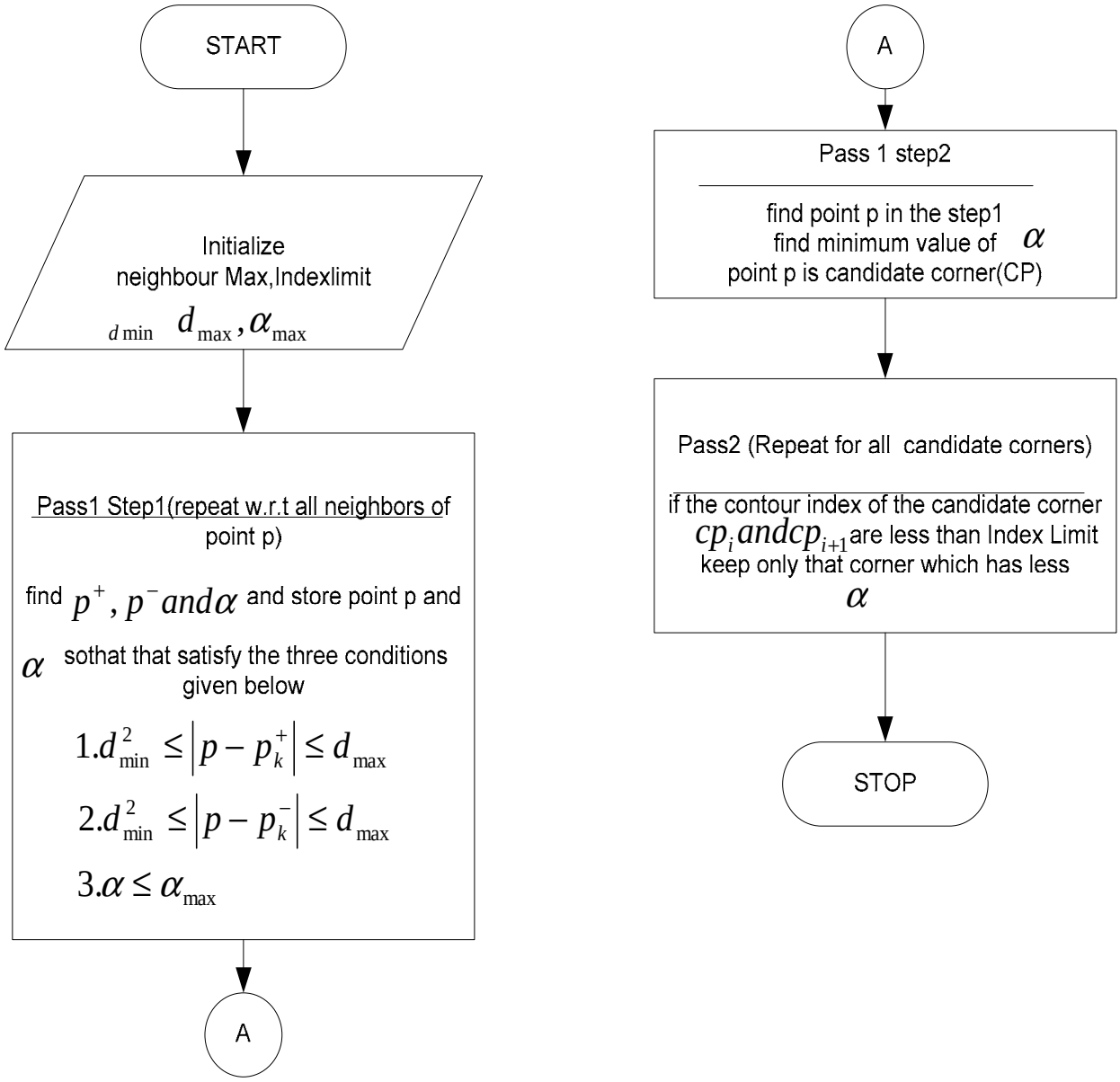


Figure 3: Corner points. $d_{min}=d_{max}=20$

Corner points are detected from the contour points using curvature analysis technique we divide the contour points of each piece into groups called segments and fit cubic Bezier curve to each segment. We have used only one piece and the number of segments is based on corner

points. It means that if there are *m* corner points then there will be *m* segments

Flow Chart for Corner Detection Algorithm



We use chord-length parameterization to estimate the parametric value *t* associated with each point p_i, which is more suitable and accurate.
After determining the Corner points, we will remove the redundant corner points within some index limit provided. For the above figure we used index limit as 10.

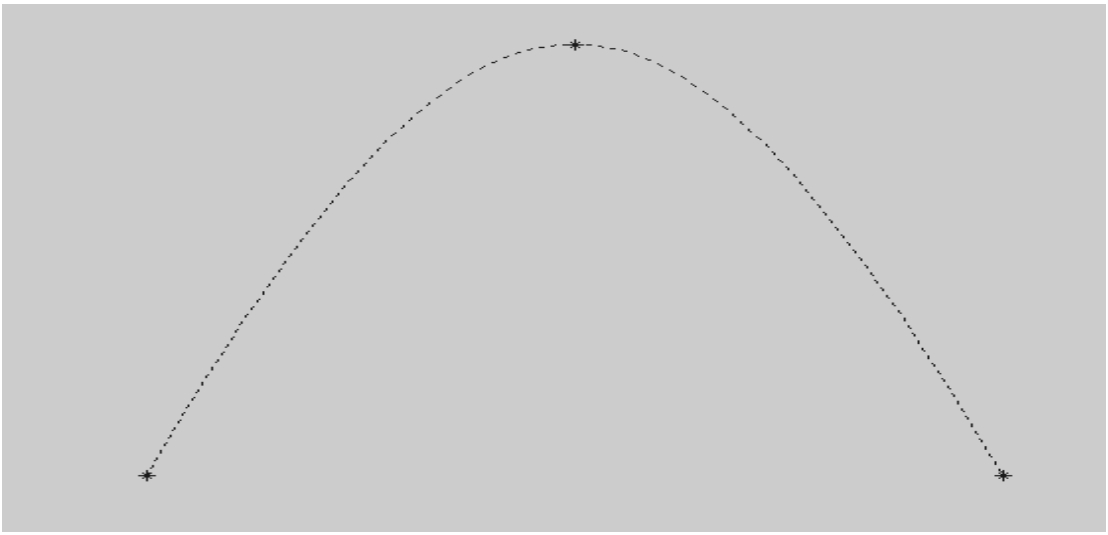


Figure 4: Corner points for index Limit=10

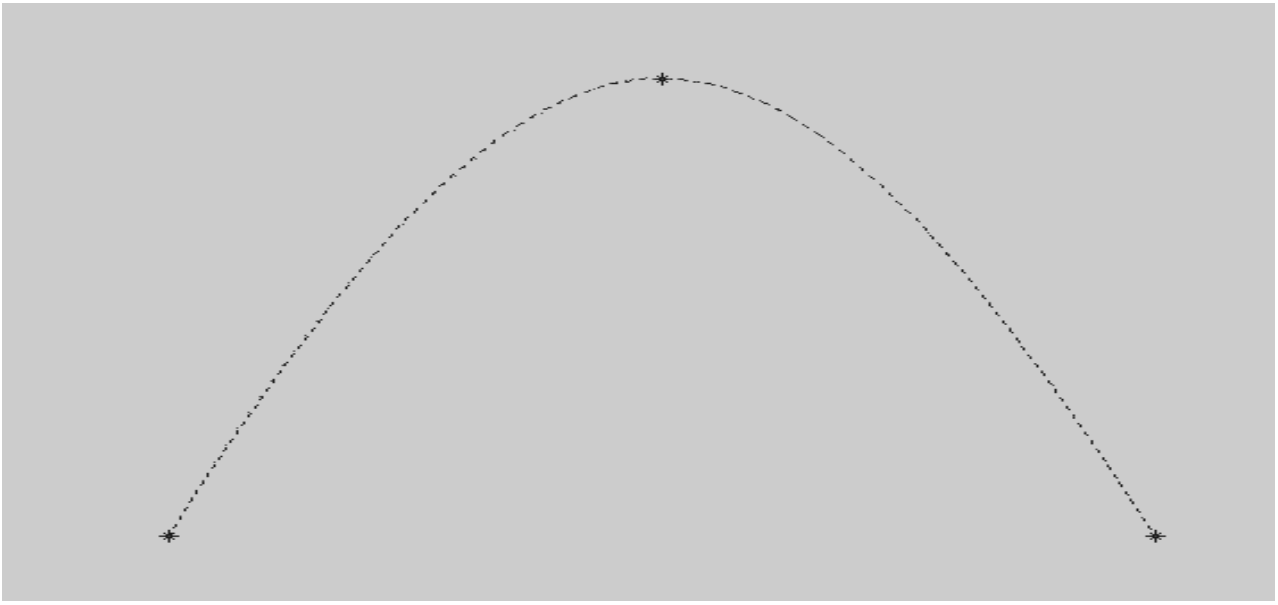


Figure 5 : Parametric Fitted Curve on Digitized Boundary

The Bezier form of cubic polynomial curve segment has four control points p_0 ,p_1 ,p_2 ,p_3 the Bezier curve interpolates the two end control points and approximates the two intermediate points. The two end control points are the two corner points of the curve segment. But we have to find the two intermediate control points of the cubic Bezier to fit the curve. For this purpose we use the least square method to determine the intermediate control points.

We fit the parametric curve for each piece after determining the intermediate control points.

$$Q_x(t) = (1-t^3)P_{x_0} + 3t(1-t^2)P_{x_1} + 3t^2(1-t)P_{x_2} + t^3P_{x_3}$$

$$Q_y(t) = (1-t^3)P_{y_0} + 3t(1-t^2)P_{y_1} + 3t^2(1-t)P_{y_2} + t^3P_{y_3}$$

Least Square Method

Our goal is to approximate the digitized curve by the parametric curve in the best way using least square method. We define the sum of the squared distances from the digitized curve to the parametric curve.
Mathematically we write:

$$\begin{aligned} S &= \sum_{i=1}^{n}[Q_i(t) - p_i]^2 \\ &= \sum_{i=1}^{n}[Q_{x_i}(t) - p_{x_i}]^2 + \sum_{i=1}^{n}[Q_{y_i}(t) - p_{y_i}]^2 \end{aligned}$$

Then our goal is to minimize the S. So, we find the partial derivative of S with respect to P_1 and P_2 and equate them to zero.

$$\begin{aligned} \frac{\partial S}{\partial P_1} &= 0 \\ \frac{\partial S}{\partial P_2} &= 0 \end{aligned}$$

Expanding the above equations:

$$\begin{aligned} \frac{\partial S}{\partial P_1} &= 2\sum_{i=1}^{n}\frac{\partial Q(t_i)}{\partial P_1}[Q(t_i) - p_i] \\ &= 2\sum_{i=1}^{n}B_1(t_i)[Q(t_i) - p_i] \end{aligned}$$

$$\frac{\partial S}{\partial P_2} = 2\sum_{i=1}^{n} \frac{\partial Q(t_i)}{\partial P_1}[Q(t_i) - p_i]$$
$$= 2\sum_{i=1}^{n} B_2(t_i)[Q(t_i) - p_i]$$

Let

$$A_k = \sum_{i=1}^{n} [B_k(t_i)]^2$$
$$A_{1,2} = \sum_{i=1}^{n} B_1(t_i)B_2(t_i)$$
$$C_{x_k} = \sum_{i=1}^{n} [B_k(t_i)[p_{x_i} - B_0(t_i)P_{x_0} - B_3(t_i)P_{x_3}]$$
$$C_{y_k} = \sum_{i=1}^{n} [B_k(t_i)[p_{y_i} - B_0(t_i)P_{y_0} - B_3(t_i)P_{y_3}]$$

Solving the above equations for P_1 and P_2 gives

$$\begin{bmatrix} A_1 & A_{1,2} \\ A_{1,2} & A_2 \end{bmatrix} \begin{bmatrix} P_{x_1} \\ P_{x_2} \end{bmatrix} = \begin{bmatrix} C_{x_1} \\ C_{x_2} \end{bmatrix}$$

$$\begin{bmatrix} A_1 & A_{1,2} \\ A_{1,2} & A_2 \end{bmatrix} \begin{bmatrix} P_{y_1} \\ P_{y_2} \end{bmatrix} = \begin{bmatrix} C_{y_1} \\ C_{y_2} \end{bmatrix}$$

P_1 and P_2 are intermediate control points. Solving the above equations we

$$P_{x_1} = \frac{A_2 C_{x_1} - A_{1,2} C_{x_2}}{A_1 A_2 - A_{1,2}^2} \qquad P_{y_1} = \frac{A_2 C_{y_1} - A_{1,2} C_{y_2}}{A_1 A_2 - A_{1,2}^2}$$

$$P_{x_2} = \frac{A_1 C_{x_2} - A_{1,2} C_{x_1}}{A_1 A_2 - A_{1,2}^2} \qquad P_{y_2} = \frac{A_1 C_{y_2} - A_{1,2} C_{y_1}}{A_1 A_2 - A_{1,2}^2}$$

The parametric curve is exactly fitted in our case. See Figure 5.

Since the parametric curve is fitted accurately, we went to some complex open curve .See Figure 6.
We applied the same technique here and fitted the parametric curve using least square method. The curve fitted for the middle segment did not accurately fitted. See Figure 10.
The fitted Bezier curve may not fit accurately always. So, to fit it more accurately we set some threshold tolerance limit. We compute the squared distance between each point p_k of digitized curve and its corresponding point $Q_k(t)$.

$$d^2_k = |pk - Qk(t)|.$$
$$= \left[p_{x_k} - \mathrm{Qk(t).}\right]^2 + \left[p_{y_k} - Q_{y_k}(t)\right]^2$$

among all computed distances we find maximum squared distance d^2_{max}

$$d^2_{max} = \mathrm{Max}(d^2_1, d^2_2, \ldots, d^2_n)$$

If d^2_{max} exceeds predefined error tolerance limit $d^2_{tolorence}$then the segment is broken into two segments at the point of maximum distance and the point corresponding to maximum distance is added to the list of significant points.
Now we fit the parametric curve for each pair of significant point(Here significant points include corner points also) using the same method mentioned earlier.

We fitted the curve for $d_{toleraance}$=5 in Figure 11.

The process is repeated for each segment until all segments of all the pieces meet the threshold tolerance limit. In each phase we decrease the

d_{max} to obtain more significant points, if the fitted curve does not accurately matches the digitized boundary.

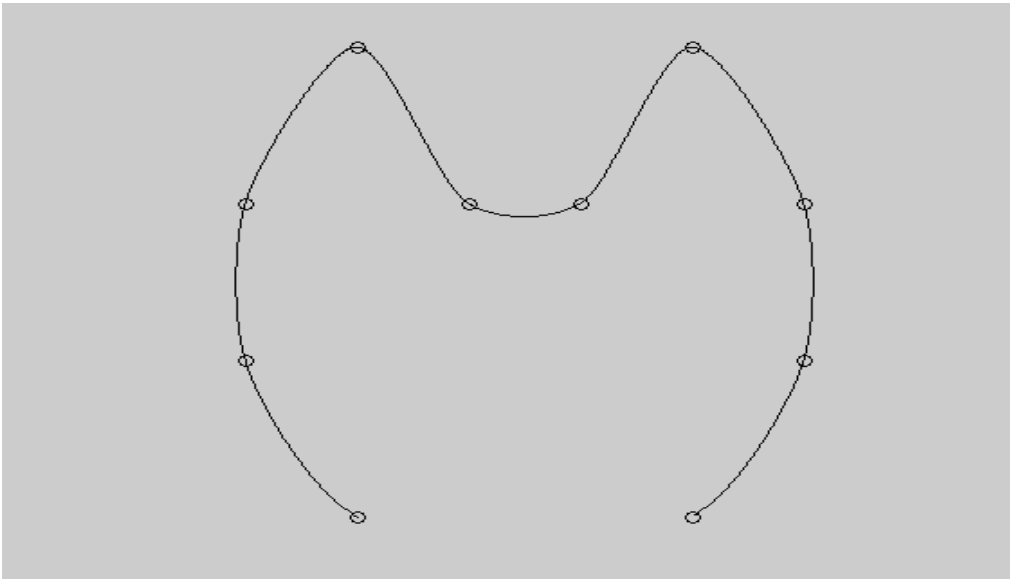


Figure 6: Open curve with more than one piece

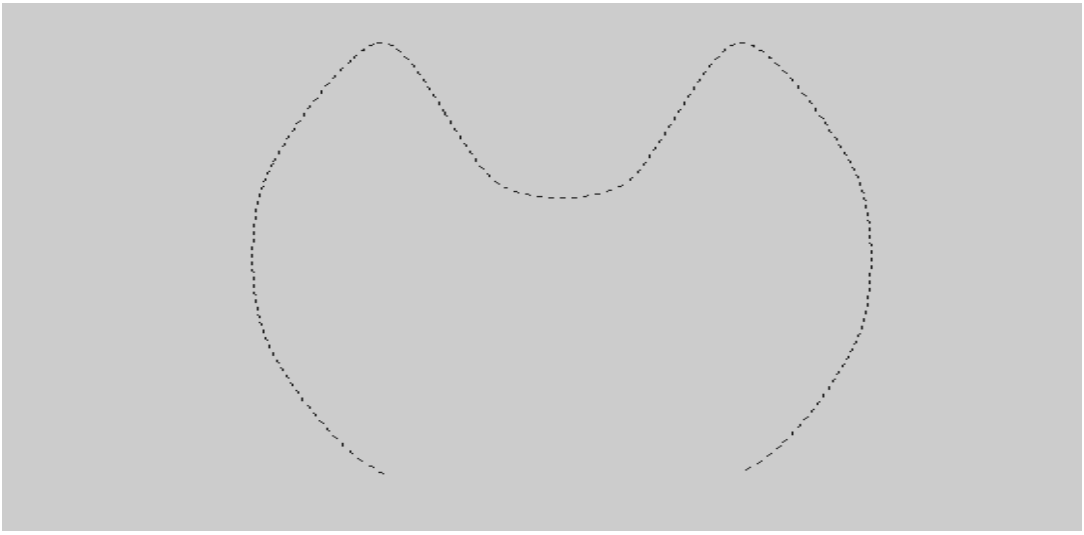


Figure 7: Digitized Boundary

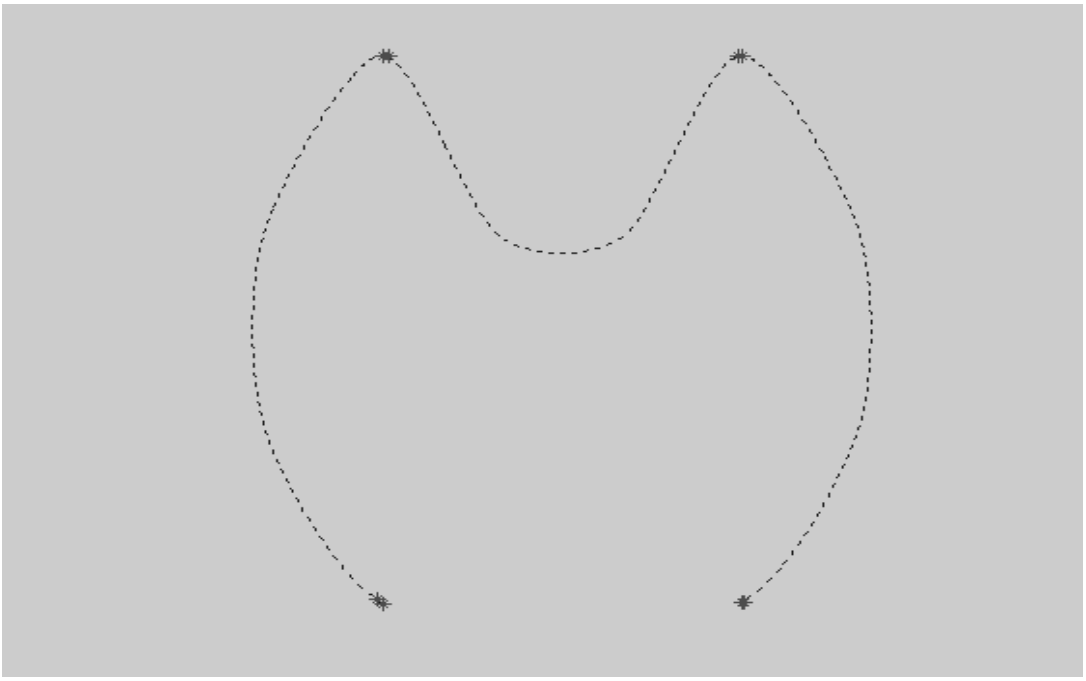


Figure 8: Corner Points d_{min}=d_{max}=20, α_{max}=30

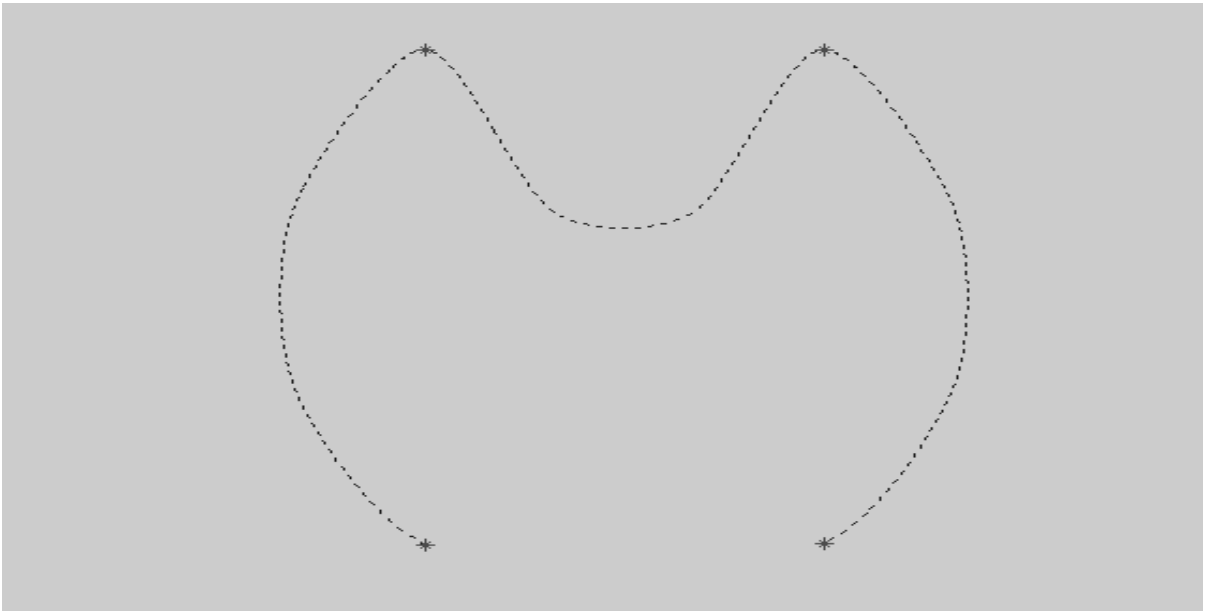


Figure 9: Redundant corner points removed

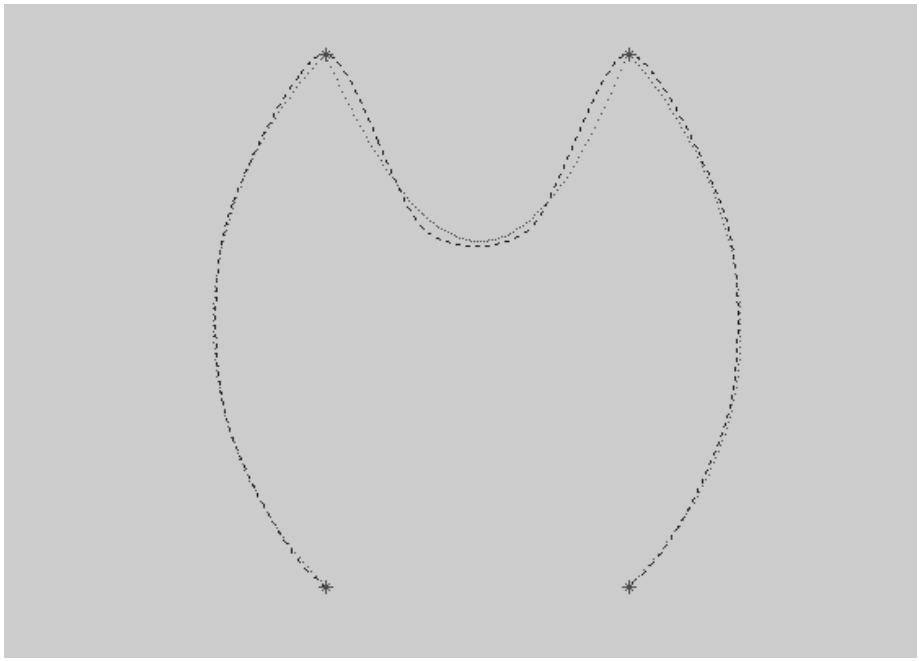


Figure 10: Parametric fitted curve over Digitized Boundary

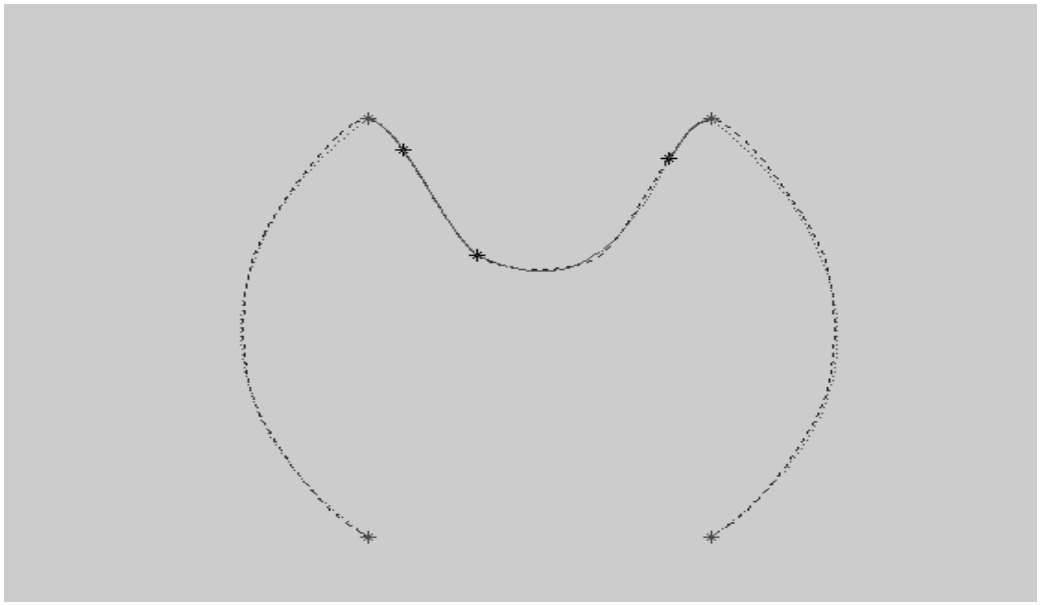
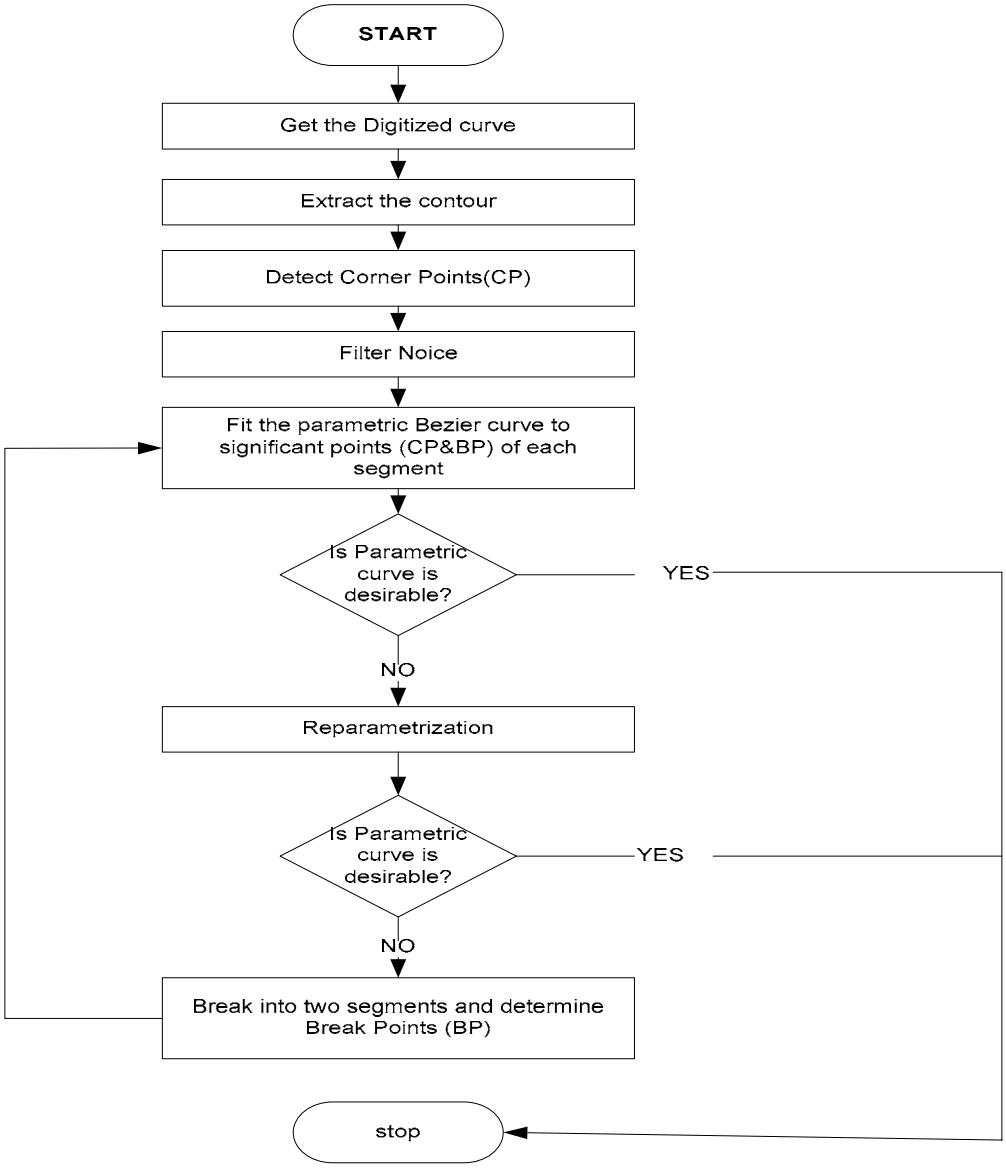


Figure 11: Parametric fitted curve after finding the breakpoints. $d_{tolerance}=5$

Flow Chart


START
Get the Digitized curve
Extract the contour
Detect Corner Points(CP)
Filter Noice
Fit the parametric Bezier curve to significant points (CP&BP) of each segment
Is Parametric curve is desirable?
YES
NO
Reparametrization
Is Parametric curve is desirable?
YES
NO
Break into two segments and determine Break Points (BP)
stop

3. Conclusion

We can use other parametric curves such as B-Spines and Hermite models.

We can extend the method so that it can work for 3D images.

We can use higher order Bezier curves if we can reduce the number of segments, but it will be expensive in terms of computation.

We can parallelize the application, if the character has more than one piece and more than one segment.